Úkoly pro lekci č.4

Závazné pokyny pro odeslání: řešení pošlete v jednom souboru s názvem `prijmeni-jmeno.rkt`, na adresu `xpap1@phoenix.inf.upol.cz`, kde v předmětu zprávy uvedete YPP1-4. V názvu přílohy nepoužívejte diakritiku.

Procedury musí mít stejné názvy a brát argumenty v pořadí, které odpovídá uvedeným testovacím výrazům. E-maily budou strojově zpracovávány.

1.Vytvořte abstrakční bariéry pro níže uvedenou reprezentaci měst
konstruktor `make-city` selektory `name` pro název, `region`, `habitants` pro počet obyvatel

```
(define cities
  (list (make-city 'Olomouc 'Stredni-Morava 100000)
        (make-city 'Prerov 'Stredni-Morava 40000)
        (make-city 'Prostejov 'Stredni-Morava 30000)
        (make-city 'Brno 'Jizni-Morava 300000)
        (make-city 'Ostrava 'Severni-Morava 350000)
        (make-city 'Praha 'Stredni-Cechy 1200000)
        (make-city 'Hradec-Kralove 'Vychodni-Cechy 100000)))
```

2.Napište bez použití `filter` proceduru `restriction`, která bude provádět operaci restrikce – t.j. filtrování řádků podle zadané procedury.

```
> (restriction (lambda (c) (equal? (region c) 'Stredni-Morava)) cities)
((Olomouc Stredni-Morava 100000)
 (Prerov Stredni-Morava 40000)
 (Prostejov Stredni-Morava 30000))
> (restriction (lambda (c) (> (habitants c) 100000)) cities)
((Brno Jizni-Morava 300000)
 (Ostrava Severni-Morava 350000)
 (Praha Stredni-Cechy 1200000))
```

3.Napište proceduru `aggregate`, která bude pomocí procedury agregovat hodnoty v daném sloupci.

```
> (define morava-cities
  (restriction (lambda (c) (or (equal? (region c) 'Jizni-Morava)
                               (equal? (region c) 'Stredni-Morava)
                               (equal? (region c) 'Severni-Morava)))
               cities))
> (aggregate + habitants morava-cities)
820000
> (aggregate min habitants morava-cities)
30000
```

4.Napište proceduru `bin2dec`, která převede seznam číslic 0 a 1 na dekadické číslo. Řešení založte na `apply`.

```
> (bin2dec '(0 1 0 1 1))
11
> (bin2dec '(0 1 1 0 1 1 1))
55
```

```
> (bin2dec '(1 1 0 0))
12
```

5.Napište proceduru `euclid`, která zjistí největšího společného dělitele dvou zadaných čísel pomocí Euklidova algoritmu pro hledání NSD. Popis viz. `http://cs.wikipedia.org/wiki/Euklidův_algoritmus`. Řešení založte na rekurzi.

```
> (euclid 12 4)
4
> (euclid 12 5)
1
> (euclid 12 6)
6
> (euclid 666 36)
18
```

6.Napište proceduru `leibniz-pi`, která provede odhad čísla pí pomocí Leibnizovy řady, popis viz `http://en.wikipedia.org/wiki/Leibniz_formula_for_pi`. Řešení založte na rekurzi.

```
> (leibniz-pi 10)
3.232315809405594
> (leibniz-pi 100)
3.1514934010709914
> (leibniz-pi 2000)
3.1420924036835256
> (leibniz-pi 20000)
3.1416426510898874
> (leibniz-pi 200000)
3.141597653564762
```

Poznámka: kvůli zaokrouhlovacím chybám vám mohou vycházet mírně odlišné výsledky odhadu čísla pí.

7.Napište proceduru `perfect?`, která zjistí zda zadané číslo je tzv. dokonalým číslem, to jest takovým, které je součtem všech svých dělitelů kromě sebe sama. Například 6 je dokonalé číslo, protože jeho děliteli jsou 3, 2 a 1, přičemž platí, že 3+2+1=6.

```
> (perfect? 6)
#t
> (perfect? 28)
#t
> (perfect? 29)
#f
> (perfect? 496)
#t
```